Mathematical Interfaces of Automated Scientific Computing

Andy R Terrel

Department of Computer Science
University of Chicago

October 21, 2009
Computing Techniques seminar
Fermilab

Acknowledgments

- L. Ridgway Scott (University of Chicago)
- Matthew R. Knepley (Argonne National Lab)
- Dmitry A. Karpeev (Argonne National Lab)
- Robert C. Kirby (Texas Tech University)
- Kevin R. Long (Texas Tech University)
- Anders Logg (Simula Research Lab)

Outline

Outline

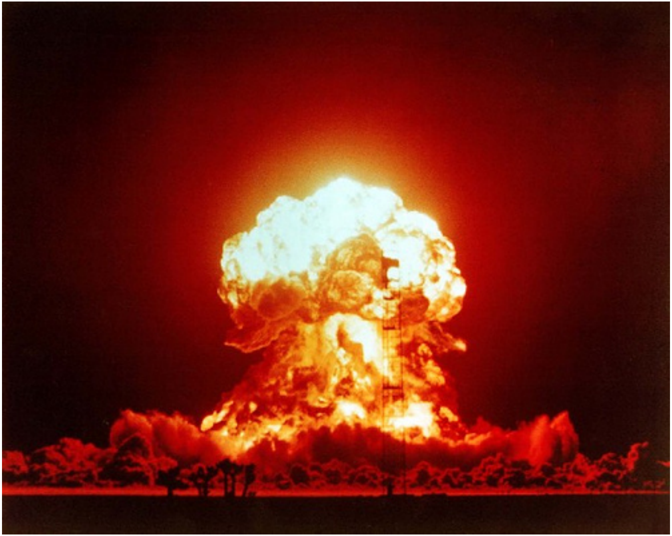
Why do simulations?

Why do simulations?

Because experiments are expensive

Why do simulations?

Because experiments are dangerous



Why do simulations?

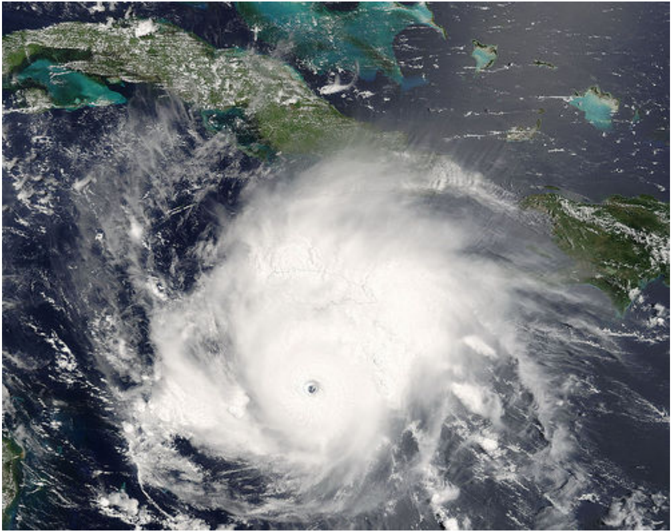
Because experiments are not possible

Why do simulations?

Because simulations are faster

Why do simulations?

Because we need the data ASAP



Simulation: Third Tier of Science

- D'Alembert's Paradox
- Supernova flashback
- Rayleigh-Taylor Constant

Simulation: Third Tier of Science

- D'Alembert's Paradox
- Supernova flashback
- Rayleigh-Taylor Constant

Simulation: Third Tier of Science

- D'Alembert's Paradox
- Supernova flashback
- Rayleigh-Taylor Constant

Simulation: Third Tier of Science

- D'Alembert's Paradox
- Supernova flashback
- Rayleigh-Taylor Constant

Big vs Little Science

The goal of this session is explore whether, when and why universities should do big or little science. Panelists may discuss why big science wastes money, exploits graduate students and makes research too short range. They may argue that little science produces results that are too deep and narrow, oblivious to global systems issues, not properly validated, and too out of touch with reality to ever be practical. Panelists may also find some advantages to both kinds of science.

ACM SIGARCH Computer Architecture News
Volume 18, Issue 3a, June 1990

Big Science and Big Computers

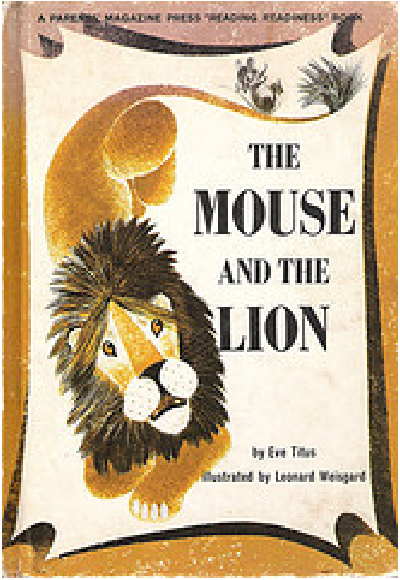
- Requires large, highly specialized coding projects
- Incredibly hard to design for maintainability, feature addition, and new hardware paradigms
- Resolves large open phenomena (or asks for more money)

Little Science and Rapid Development

- Able to use inefficient (general) methods
- Usually only test on small problems
- Can use (somewhat) exhaustive search of different possible methods.
- High Productivity Environment

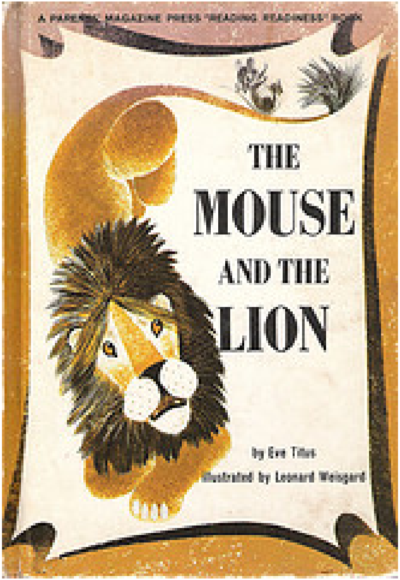
Automation becomes the Thorn

- Pervasive abstractions
- Write general code, Generate specific code
- Fails due to bad interfaces



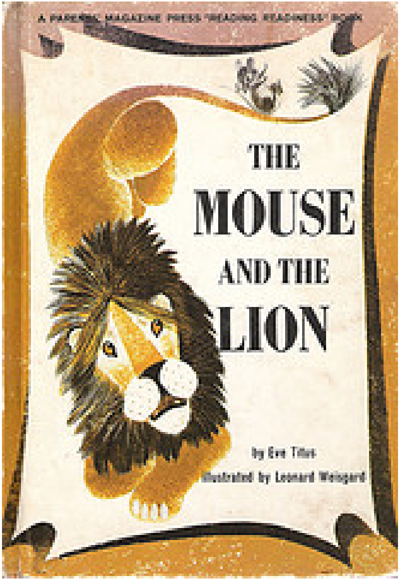
Automation becomes the Thorn

- Pervasive abstractions
- Write general code, Generate specific code
- Fails due to bad interfaces



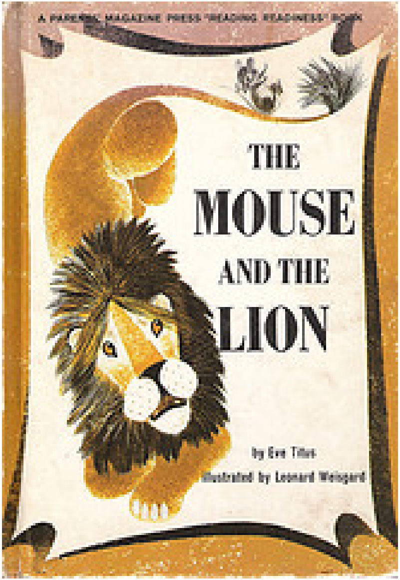
Automation becomes the Thorn

- Pervasive abstractions
- Write general code, Generate specific code
- Fails due to bad interfaces



Automation becomes the Thorn

- Pervasive abstractions
- Write general code, Generate specific code
- Fails due to bad interfaces



Outline

The Productivity Factors

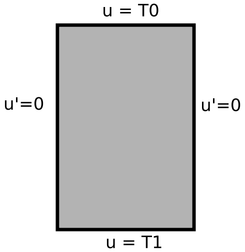
How much code do I have to write:

Written Code	Generated Code
ANSI C: 50 lines	Assembler: 200 lines
FFC: 10 lines	C++: 20K lines
Quantum Chemistry: 6 symbols	FORTRAN: 1M lines

The FEM-PDE model

Find u on domain Ω, given f and BC

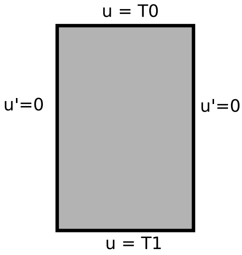
$$-\Delta u = f$$



The FEM-PDE model

Find u on domain Ω, given f and BC,
such that for all v in the function space S

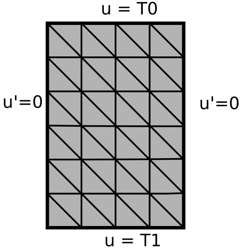
$$a(u,v) = (f,v)$$



The FEM-PDE model

Find u_h on a triangulization of domain Ω,
given f and BC,
such that for all v in the function space S

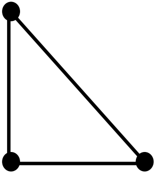
$$a(u_h, v) = (f, v)$$

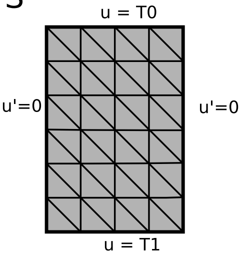


The FEM-PDE model

Find u_h on a triangulization of domain Ω,
given f and BC,
such that for all v_h
in the function space V ⊂ S

a(u_h,v_h) = (f,v_h)

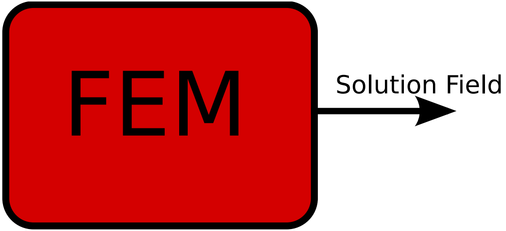




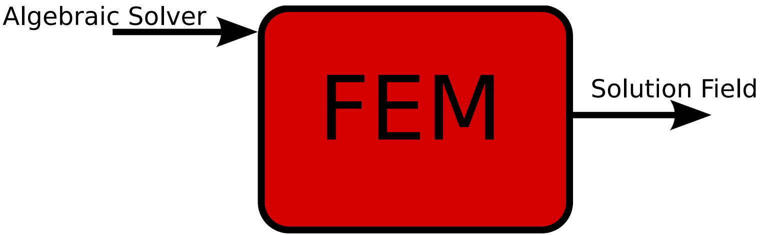
The FEM-PDE model

$$\left[\quad A \quad\right]\left[\, x \,\right] = \left[\, b \,\right]$$

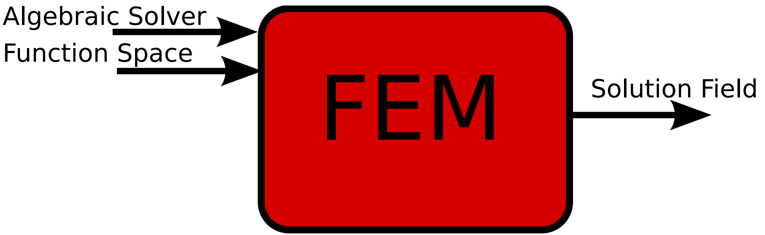
The FEM-PDE model



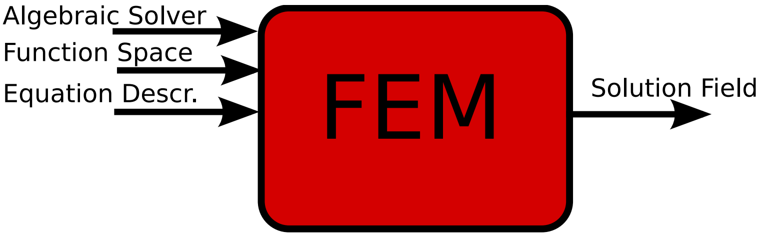
The FEM-PDE model



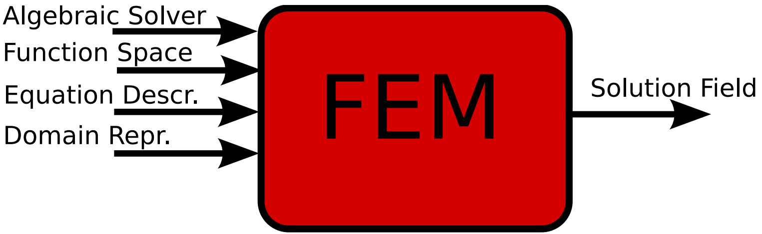
The FEM-PDE model



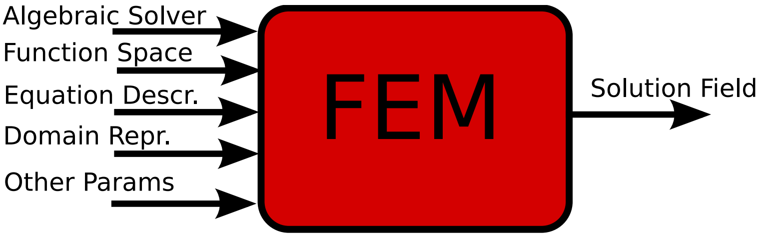
The FEM-PDE model



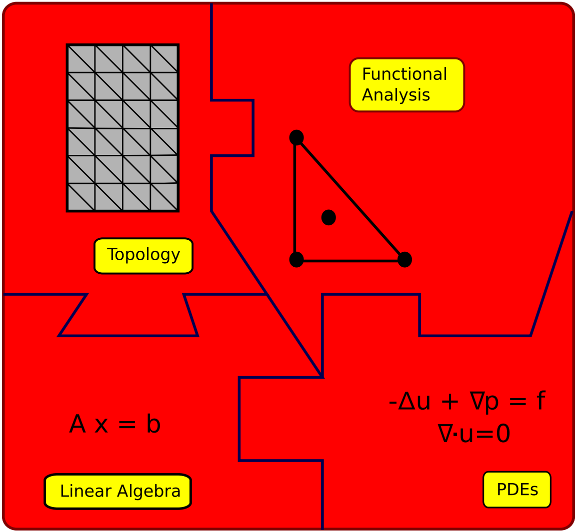
The FEM-PDE model



The FEM-PDE model



Mathematics Necessary



Algebraic Solvers

Why use Ax=b?

$$\begin{bmatrix} \quad & A & \quad \end{bmatrix}\begin{bmatrix} x \end{bmatrix} = \begin{bmatrix} b \end{bmatrix}$$

- Model is able to capture lots of computations
- Reisz Representation Theorem

The Large Scale Success Story

- BLAS
- LAPACK
- Scalapack
- Atlas
- Flame
- Trilinos
- PETSc
- Hypre
- ... More to come (Salsa)

Functional Spaces

Function Space Matters

Stokes Equation

- Taylor-Hood
- Crouzeix-Raviart
- Iterated Penalty

$$\begin{aligned} -\Delta \mathbf{u} + \nabla \mathbf{p} &= f \\ \nabla \cdot \mathbf{u} &= 0 \end{aligned}$$

Function Space Matters

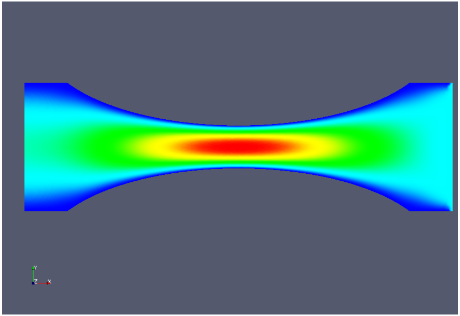
$$\frac{du}{dt} + u \cdot \nabla u = -\frac{\nabla \mathbf{p}}{\rho} + \nu \Delta \mathbf{u}$$

Stokes Equation
Taylor-Hood
Crouzeix-Raviart
Iterated Penalty

Navier-Stokes

- Stokes Solver
- Nonlinear Solver
- Time Stepping

Function Space Matters



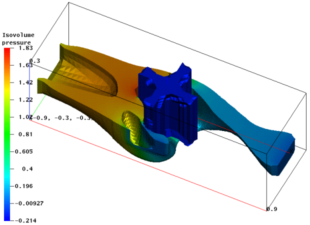
Navier-Stokes
Stokes Solver
Nonlinear Solver
Time Stepping

Stokes Equation
Taylor-Hood
Crouzeix-Raviart
Iterated Penalty

Non-Newtonian Flow

- Oldroyd-B
- Grade 2

Function Space Matters



Stokes Equation
Taylor-Hood
Crouzeix-Raviart
Iterated Penalty

Navier-Stokes
Stokes Solver
Nonlinear Solver
Time Stepping

Non-Newtonian
Odroyd-B
Grade 2
...

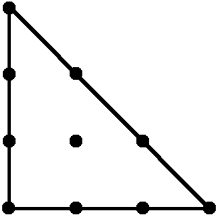
Fluid Solid Interfaces

- Free Boundary Problems
- Couple to legacy Codes

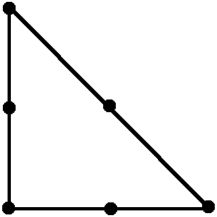
Success Story

- FIAT Algorithm [Kirby 2005]
- Syfi [Mardel et al 2007]

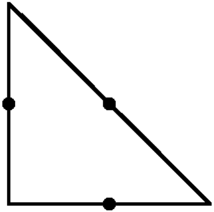
Stokes Function Spaces



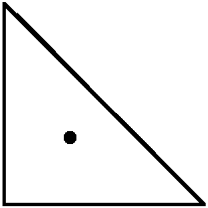
(a) P_3 for V



(b) P_2 for Π



(c) Crouzeix-Raviart for V

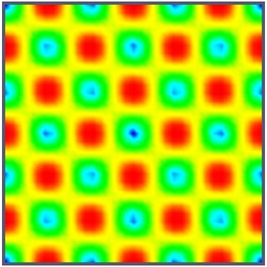


(d) P_0 for Π

Problem statement

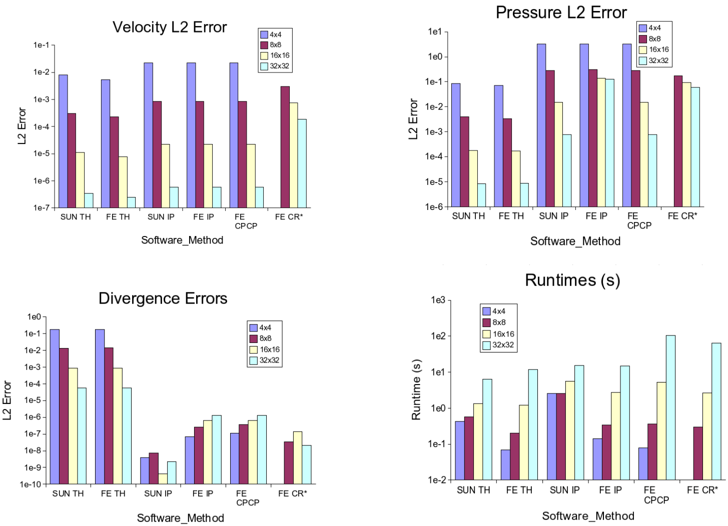
$$\mathbf{u} = \begin{bmatrix} \sin(3\pi x)\cos(3\pi y) \\ -\cos(3\pi x)\sin(3\pi y) \end{bmatrix}$$

$$p = \sin(3\pi x)\sin(3\pi y)$$



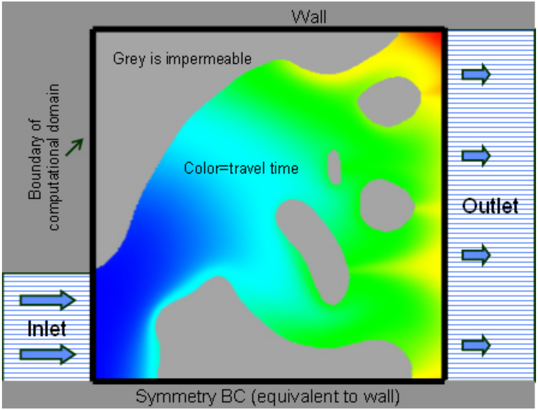
Important Numbers.

Comparison of Fourth Order

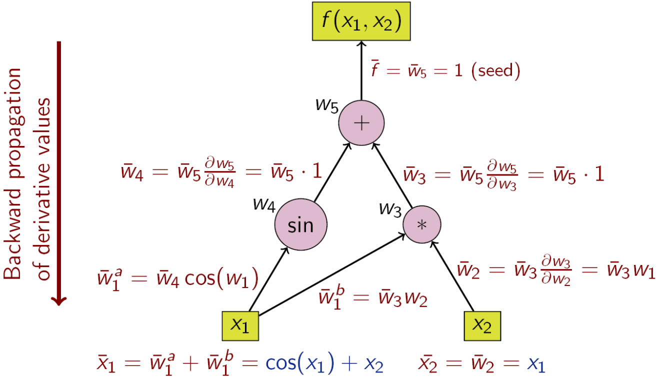


Equation Description

Optimization



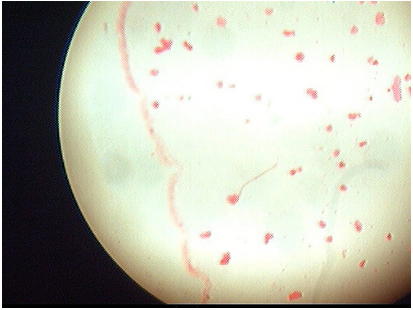
Automatic Differentiation



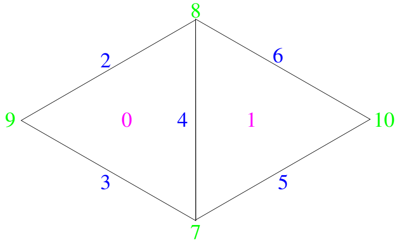
Domain Representation

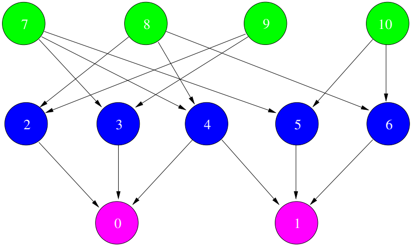
Two Applications

Two Applications

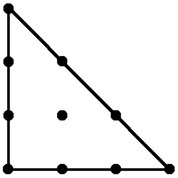


Sieve





Easy General Mesh



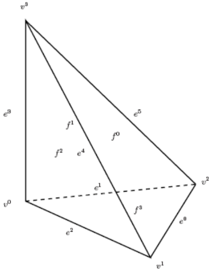
Simple Mesh

Points: 1,2,3
Edges: (1,2),(1,3),(2,3)
Face: (1,2,3)

Sieve Mesh

Points: 1,2,3
Edges: support(Points)
Face: support(Edges)

Easy General Mesh



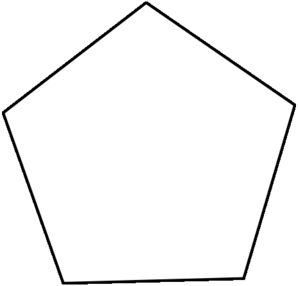
Simple Mesh

Points: 1,2,3,4
Edges: (1,2),(1,3),
(1,4),(2,3),(2,4),(3,4)
Face: (1,2,3),(1,2,4),
(1,3,4),(2,3,4)

Sieve Mesh

Points: 1,2,3,4
Edges: support(Points)
Faces: support(Edges)

Easy General Mesh



Sieve Mesh

Points: 1,2,3,4,5
Edges: support(Points)
Faces: support(Edges)

Simple Mesh

Unsupported.

Outline

Automation Standard

Already Matlab is standard. Why?

Automation Standard

Already Matlab is standard. Why?

Because with '\', the user does not have to chose between the following algorithms:

- Cholesky factorization
- QR factorization
- LU factorization
- Gaussian elimination with partial pivoting
- Least Squares fitting

Computing = Big Computing

We should not settle for less

Questions

Andy R Terrel
Computer Science Department
University of Chicago, Chicago, IL
aterrel@uchicago.edu